

LOG4J - LOGGING METHODS

http://www.tutorialspoint.com/log4j/log4j_logging_methods.htm

Copyright © tutorialspoint.com

Logger class provides a variety of methods to handle logging activities. The Logger class does not allow us to instantiate a new Logger instance but it provides two static methods for obtaining a Logger object:

- **public static Logger getLogger();**
- **public static Logger getLogger(String name);**

Here the first of the two methods returns the application instance's root logger and does not have a name. Any other named Logger object instance is obtained through the second method by passing the name of the logger. The name of the logger can be any string you can pass, usually class or package name as we have used in last chapter and it is mentioned below:

```
static Logger log = Logger.getLogger(log4jExample.class.getName());
```

Logging Methods:

Once we obtain an instance of a named logger, we can use several methods of the logger to log messages. The Logger class has the following methods for printing the logging information.

SN	Methods with Description
1	public void debug(Object message) This method prints messages with the level Level.DEBUG.
2	public void error(Object message) This method prints messages with the level Level.ERROR.
3	public void fatal(Object message); This method prints messages with the level Level.FATAL.
4	public void info(Object message); This method prints messages with the level Level.INFO.
5	public void warn(Object message); This method prints messages with the level Level.WARN.
6	public void trace(Object message); This method prints messages with the level Level.TRACE.

All the levels are defined in the org.apache.log4j.Level class and any of the above mentioned method can be called as follows:

```
import org.apache.log4j.Logger;

public class LogClass {
    private static org.apache.log4j.Logger log = Logger
        .getLogger(LogClass.class);

    public static void main(String[] args) {
        log.trace("Trace Message!");
        log.debug("Debug Message!");
    }
}
```

```
log.info("Info Message!");  
log.warn("Warn Message!");  
log.error("Error Message!");  
log.fatal("Fatal Message!");  
}  
}
```

When you compile and run LogClass program it would generate following result:

```
Debug Message!  
Info Message!  
Warn Message!  
Error Message!  
Fatal Message!
```

All the debug messages make more sense when they are used in combination of levels. I would cover level in next chapter and then you would have good understanding on how to use these methods in combination of different levels of debugging.